TECHNISCHE UNIVERSITÄT DRESDEN

REWERSE reasoning on the web

Hybrid integration of rules and ontologies:

A constraint-based framework

Jakob Henriksson, Jan Małuszyński

RuleML 2006, Athens, GA

The objective

- Define a scheme that

  from given
  - Rule language **R** (e.g. Datalog, Xcerpt)
  - Logical language **S** (e.g. OWL-DL, ...)

  constructs
  - A language $\mathbf{R_S}$ integrating R and S:
    - Syntax, Semantics of R_S: from syntax and semantics of R and S
    - A (complete) reasoner for R_S by interfacing *existing* reasoners of R and S

TECHNISCHE UNIVERSITÄT DRESDEN

Outline

- Motivating example
- The scheme
  - Principles and restrictions
- Reusing reasoners
  - Datalog + OWL-DL
  - Xcerpt + OWL-DL
- Practical reasoning
  - Eager interaction
- Conclusions
- Future work

Motivating example

Rule component Π:

r_1: price-in-usa(X,high) :-
made-by(X,Y),
NoFellowCompany(Y).

r_2: price-in-usa(X,high) :-
made-by(X,Y),
AmericanAssociate(Y),
monopoly-in-usa(Y,X).

r_3: made-by(a,b).
r_4: monopoly-in-usa(b,a).

DL component Σ:

T-Box:

$$\begin{aligned}
\text{European} \cap \text{American} &\subseteq \bot \\
\text{NoFellowCompany} &\equiv \forall \text{associate}.\neg\text{American} \\
\text{EuropeanAssociate} &\equiv \exists \text{associate}.\text{American} \\
\text{AmericanAssociate} &\equiv \exists \text{associate}.\text{American} \\
\text{InternationalCompany} &\equiv \text{EuropeanAssociate} \cup \text{AmericanAssociate}
\end{aligned}$$

A-Box:

InternationalCompany(b)

Ref: A.Levy and M C.Rousset. *CARIN:A Representation Language Combining Horn rules and Description Logics*. Artificial Intelligence 104(1 2):165 –209, 1998.

TECHNISCHE UNIVERSITÄT DRESDEN

Motivating example

Rule component Π:

r_1: price-in-usa(X,high) :-
made-by(X,Y),
NoFellowCompany(Y).

r_2: price-in-usa(X,high) :-
made-by(X,Y),
AmericanAssociate(Y),
monopoly-in-usa(Y,X).

r_3: made-by(a,b).
r_4: monopoly-in-usa(b,a).

DL component Σ:

T-Box:

European $\cap$ American $\subseteq$ $\perp$
NoFellowCompany $\equiv$ $\forall$associate.$\neg$American
EuropeanAssociate $\equiv$ $\exists$associate.American
AmericanAssociate $\equiv$ $\exists$associate.American
InternationalCompany $\equiv$ EuropeanAssociate $\cup$ AmericanAssociate

A-Box:

InternationalCompany(b)

Constraining extent of head predicates with ...

... constraint domain.

Motivating example

Rule component Π:

r_1: price-in-usa(X,high) :-
made-by(X,Y),
NoFellowCompany(Y).

r_2: price-in-usa(X,high) :-
made-by(X,Y),
AmericanAssociate(Y),
monopoly-in-usa(Y,X).

r_3: made-by(a,b).
r_4: monopoly-in-usa(b,a).

DL component Σ:

T-Box:

$\text{European} \cap \text{American} \subseteq \bot$
$\text{NoFellowCompany} \equiv \forall \text{associate}.\neg\text{American}$
$\text{EuropeanAssociate} \equiv \exists \text{associate}.\text{American}$
$\text{AmericanAssociate} \equiv \exists \text{associate}.\text{American}$
$\text{InternationalCompany} \equiv \text{EuropeanAssociate} \cup \text{AmericanAssociate}$

A-Box:

InternationalCompany(b)

$$\Pi \cup \Sigma \models \text{price-in-usa(a, high)} \;?$$

Motivating example

Rule component Π:

r_1: price-in-usa(a,high) :-
made-by(a,b),
NoFellowCompany(b).

r_2: price-in-usa(X,high) :-
made-by(X,Y),
AmericanAssociate(Y),
monopoly-in-usa(Y,X).

r_3: made-by(a,b).
r_4: monopoly-in-usa(b,a).

DL component Σ:

T-Box:

European $\sqcap$ American	$\sqsubseteq \bot$
NoFellowCompany	$\equiv \forall$associate.$\neg$American
EuropeanAssociate	$\equiv \exists$associate.American
AmericanAssociate	$\equiv \exists$associate.American
InternationalCompany	$\equiv$ EuropeanAssociate $\sqcup$ AmericanAssociate

A-Box:

InternationalCompany(b)

$\Sigma \not\models$ NoFellowCompany(b)

$\Pi \cup \Sigma \models$ price-in-usa(a, high) ?

Motivating example

Rule component Π:

r_1: price-in-usa(X,high) :-
made-by(X,Y),
NoFellowCompany(Y).

r_2: price-in-usa(a,high) :-
made-by(a,b),
AmericanAssociate(b),
monopoly-in-usa(b,a).

r_3: made-by(a,b).
r_4: monopoly-in-usa(b,a).

DL component Σ:

T-Box:

European ⊓ American ⊑ ⊥
NoFellowCompany ≡ ∀associate.¬American
EuropeanAssociate ≡ ∃associate.American
AmericanAssociate ≡ ∃associate.American
InternationalCompany ≡ EuropeanAssociate ⊔ AmericanAssociate

A-Box:

InternationalCompany(b)

$\Sigma \not\models$ AmericanAssociate(b)

$\Pi \cup \Sigma \models$ price-in-usa(a, high) ?

Motivating example

Rule component Π:

r_1: price-in-usa(a,high) :-
made-by(a,b),
NoFellowCompany(b).

r_2: price-in-usa(a,high) :-
made-by(a,b),
AmericanAssociate(b),
monopoly-in-usa(b,a).

r_3: made-by(a,b).
r_4: monopoly-in-usa(b,a).

DL component Σ:

T-Box:

European ⊓ American ⊑ ⊥
NoFellowCompany ≡ ∀associate.¬American
EuropeanAssociate ≡ ∃associate.American
AmericanAssociate ≡ ∃associate.American
InternationalCompany ≡ EuropeanAssociate ⊔ AmericanAssociate

A-Box:

InternationalCompany(b)

But:

$\Sigma \models$ NoFellowCompany(b)
∨
AmericanAssociate(b)

$\Pi \cup \Sigma \models$ price-in-usa(a, high) ?

Motivating example

Rule component Π:

r_1: price-in-usa(X,high) :-
made-by(X,Y),
NoFellowCompany(Y).

r_2: price-in-usa(X,high) :-
made-by(X,Y),
AmericanAssociate(Y),
monopoly-in-usa(Y,X).

r_3: made-by(a,b).
r_4: monopoly-in-usa(b,a).

DL component Σ:

T-Box:

$\text{European} \cap \text{American} \subseteq \bot$
$\text{NoFellowCompany} \equiv \forall \text{associate}.\neg\text{American}$
$\text{EuropeanAssociate} \equiv \exists \text{associate}.\text{American}$
$\text{AmericanAssociate} \equiv \exists \text{associate}.\text{American}$
$\text{InternationalCompany} \equiv \text{EuropeanAssociate} \cup \text{AmericanAssociate}$

A-Box:

InternationalCompany(b)

Thus, $\Pi \cup \Sigma \models$ price-in-usa(a, high) !

TECHNISCHE UNIVERSITÄT DRESDEN

Rules we consider

HEAD ← BODY

- HEAD is some basic construct (*atom*)
- BODY is a set of atoms
- **Safety**: head variables appear in the body
- Examples:
  - Datalog: atomic formulae
  - Xcerpt: *Query terms* and *Construct terms*

Semantics of rules

- Fixpoint semantics
  - Rules derive ground atoms from given ground atoms
    - ***matching*** of body atoms vs. given atoms gives substitution θ
    - θ applied to head ➔ derived atom

$$T_P(S) = \{ H\theta \mid (H \leftarrow B_1, \ldots, B_n) \in P \text{ and } (B_1, \ldots, B_n) \textit{ matches} \text{ some } A_1, \ldots, A_n \text{ in } S \text{ with result } \theta \}$$

  - T_P monotonic, $T_P(S) \subset T_P(S')$ for any $S \subset S'$
  - **Semantics of program P**: least fixpoint of T_P

Examples of rules languages

The class includes

- Logical rule languages, e.g.
  - **Datalog** (without negation)
  - Semantics of program: set of Datalog atoms
  - Least Herbrand model
- Rule languages lacking logical semantics, e.g.
  - **Xcerpt** (negation-free subset)
  - Semantics of program: set of Xcerpt data terms

Extended rules

$$\text{HEAD} \leftarrow \text{BODY},\mathbf{C}$$

- **C** formula of an ***external theory*** in logical language ***L***
- Ground atoms associated with a constraint
  - **A;C** where **A** is a ground atom, **C** formula of ***L***
- Extend T_P operator:

$$\begin{aligned} T_P(S) = \{ & H\theta; (C\theta \wedge C_1 \wedge \ldots \wedge C_n) \mid (H \leftarrow B_1, \ldots, B_n, C) \in P \text{ and} \\ & \text{for some } A_1;C_1, \ldots, A_n;C_n \text{ in } S \\ & (B_1, \ldots, B_n) \textit{ matches } A_1, \ldots, A_n \text{ with result } \theta \} \end{aligned}$$

Semantics of extended rules

- Restrict model of underlying rule program
  - A *constraint* C, wrt. an external theory Σ, can be:

    1. True in all models of Σ ($\Sigma \models C$)
    2. False in all models of Σ ($\Sigma \models \neg C$)
    3. None of above:
       satisfiable, but false in some models of Σ

$$M(P) = \{ A \mid A \in lfp(T_P) \text{ and } \Sigma \models C_A \}$$

- C_A is a disjunction of all constraints of A

Example instances

- Existing rule reasoners not aware of “external” predicates
  - How to re-use rule reasoners to collect constraints?
  - Solved specifically for each language and rule reasoner

- Here:

(1) Datalog + OWL DL

(2) Xcerpt + OWL DL

(1) Collecting constraints in XSB for Datalog

Π

```
r1: price-in-usa(X,high) :-
      made-by(X,Y),
      NoFellowCompany(Y).

r2: price-in-usa(X,high) :-
      made-by(X,Y),
      AmericanAssociate(Y),
      monopoly-in-usa(Y,X).

r3: made-by(a,b).
r4: monopoly-in-usa(b,a).
```

→

Π'

```
r1: price-in-usa(X,high,[NoFellowCompany(Y) | A]) :-
      made-by(X,Y,A).

r2: price-in-usa(X,high,[AmericanAssociate(Y) | A]) :-
      made-by(X,Y,A1),
      monopoly-in-usa(Y,X,A2),
      append(A1,A2,A).

r3: made-by(a,b,[]).
r4: monopoly-in-usa(b,a,[]).
```

TECHNISCHE UNIVERSITÄT DRESDEN

(1) Collecting constraints in XSB for Datalog

- **Query ← *price-in-usa(a,high,C)* wrt. Π':**

C = [NoFellowCompany(b)]
C = [AmericanAssociate(b)]

Π'

```
r1: price-in-usa(X,high,[NoFellowCompany(Y) | A]) :-
      made-by(X,Y,A).

r2: price-in-usa(X,high,[AmericanAssociate(Y) | A]) :-
      made-by(X,Y,A1),
      monopoly-in-usa(Y,X,A2),
      append(A1,A2,A).

r3: made-by(a,b,[]).
r4: monopoly-in-usa(b,a,[]).
```

ground(Π)

```
r1: price-in-usa(a,high) :-
      made-by(a,b),
      NoFellowCompany(b).

r2: price-in-usa(a,high) :-
      made-by(a,b),
      AmericanAssociate(b),
      monopoly-in-usa(b,a).

r3: made-by(a,b).
r4: monopoly-in-usa(b,a).
```

TECHNISCHE UNIVERSITÄT DRESDEN

(2) Collecting constraints in Xcerpt

```
GOAL
  prices [ all high [ var Product ] ]
FROM
  results {{
    or {
      madeby [ var Product, var M ],
      monopoly [ var M, var Product ] }
  }}
END
```

Π

```
CONSTRUCT
  results { all madeby {
    var Product, var Manufact
  } }
FILTER
  NoFellowCompany { var Manufact }
FROM
  madeby  {
      product { var Product },
      manufacturer { var Manufact }
  }
END
```

```
CONSTRUCT
  results { all monopoly {
    var Manufact, var Product
  } }
FILTER
  AmericanAssociate { var Manufact }
FROM
  and {
    monopoly {
      name { var Manufact },
      product { var Product }
    },
    madeby {
      product { var Product },
      manufacturer { var Manufact }
    }
END
```

made-by [product [“A”], manufacturer [“B”]] **XML data** monopoly [name [“B”], product [“A”]]

TECHNISCHE UNIVERSITÄT DRESDEN

(2) Collecting constraints in Xcerpt

```
GOAL
  prices [ all high [ var Product ] ]
FROM
  results {{
    or {
      r
      r
  }}
END
```

Π′

```
CONSTRUCT
  results [ all madeby [
    var Product, var Manufact,
    constraint [ instance [
      ind [ var Manufact ],
      catom [ "NoFellowCompany" ] ] ]
  ] ]
FROM
  madeby  {
      product { var Product },
      manufacturer { var Manufact }
  }
END
```

```
...uct
...
Manufact }
...
  and {
    monopoly {
      name { var Manufact },
      product { var Product }
    },
    madeby {
      product { var Product },
      manufacturer { var Manufact }
    }
END
```

Π

```
CONSTRUCT
  results { all madeby {
    var Product, var Manufact
  } }
FILTER
  NoFellowCompany { var Manufact }
FROM
  madeby  {
      product { var Product },
      manufacturer { var Manufact }
  }
END
```

made-by [product [“A”], manufacturer [“B”]] **XML data** monopoly [name [“B”], product [“A”]]

```
GOAL
  prices [ all high [ var Product ] ]
FROM
  results {{
    or {
      madeby [ var Product, var C1 ],
      ...opoly [ var C2, var Product ]
```

(2)

Π'

```
CONSTRUCT
  results [
    all monopoly [
      var Manufact, var Product,
      constraint [ instance [
        ind [ var Manufact ],
        catom [ "AmericanAssociate" ] ] ]
    ]
  ]
FROM
  and {
    monopoly {
      name { var Manufact },
      product { var Product }
    },
    madeby {
      product { var Product },
      manufacturer { var Manufact }
    }
  }
END
```

```
CONSTRU...
  results ...
    var Pro...
  } }
FILTER
  NoFello...
FROM
  madeby...
      prod...
      manu...
  }
END
```

Π

```
CONSTRUCT
  results { all monopoly {
    var Manufact, var Product
  } }
FILTER
  AmericanAssociate { var Manufact }
FROM
  and {
    monopoly {
      name { var Manufact },
      product { var Product }
    },
    madeby {
      product { var Product },
      manufacturer { var Manufact }
    }
  }
END
```

made-by [product [“A”], manufacturer [“B”]] **XML data** monopoly [name [“B”], product [“A”]]

TECHNISCHE UNIVERSITÄT DRESDEN

(2) Collecting constraints in Xcerpt

Π

```
GOAL
  prices [ all high [ var Product ] ]
FROM
  results {{
    or {
      madeby [ var Product, var M ],
      monopoly [ var M, var Product ] }
  }}
END
```

Π'

```
GOAL
  prices [
    all high [ var Product, digor [ all var C ] ] ]
FROM
  results {{
    or {
      madeby [ var Product, var M, var C ],
      monopoly [ var M, var Product, var C ]  }
  }}
END
```

```
CONSTR
  results
    var Pr
  } }
FILTER
  NoFell
FROM
  madeby  {
      product { var Product },
      manufacturer { var Manufact }
  }
END
```

```
CONSTRUCT
  results { all monopoly {
    var Manufact, var Product
  } }
FILTER
  AmericanAssociate { var Manufact }
FROM
  and {
    monopoly {
      name { var Manufact },
      product { var Product }
    },
    madeby {
      product { var Product },
      manufacturer { var Manufact }
    }
  }
END
```

made-by [product [“A”], manufacturer [“B”]] **XML data** monopoly [name [“B”], product [“A”]]

Answer from Π'

(2) Collecting constraints in Xcerpt

Query wrt. Π':

```
GOAL
  prices [
    all high [ var Product, digor [ all var C ] ] ]
FROM
  results {{
    or {
      madeby [ var Product, var M, var C ],
      monopoly [ var M, var Product, var C ]  }
  }}
END
```

$\Sigma \models$ NoFellowCompany(B) v AmericanAssociate(B)

```
prices [
        ⋮
  high [
    "A",
    digor [
      constraint [
        instance [
          ind [
            name [
              "B"
            ] ],
          catom [
            name [
              "NoFellowCompany"
            ] ] ] ],
      constraint [
        instance [
          ind [
            name [
              "B"
            ] ],
          catom [
            name [
              "AmericanAssociate"
            ] ] ] ] ]
  ],
        ⋮
]
```

TECHNISCHE UNIVERSITÄT DRESDEN

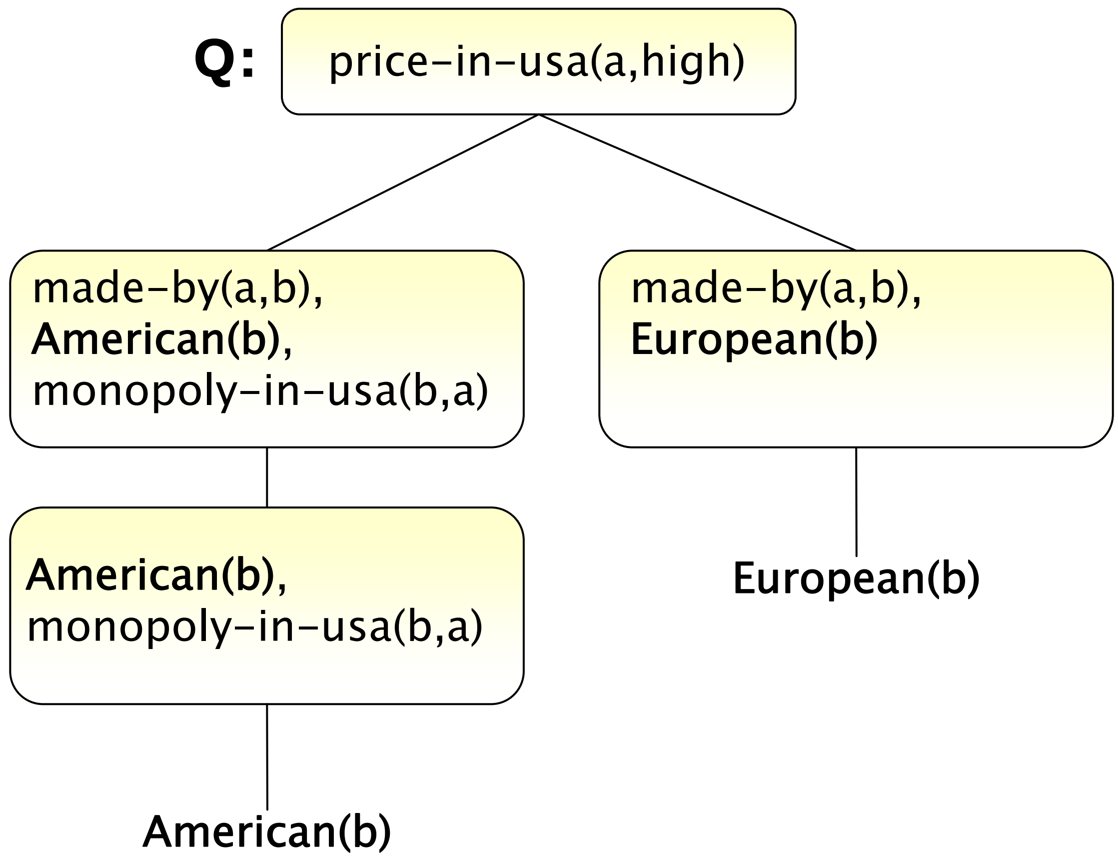
Implementing integrated reasoner with Xcerpt

1. Compile **Extended Xcerpt** (Π) programs into plain Xcerpt (Π')

$$\Pi \rightarrow \Pi'$$

   - Collect all constraints related to an Xcerpt answer using the **all** construct
2. Run Π' in existing Xcerpt engine, returning answers and (boolean) DL queries (in DIG syntax)
3. Submit DL queries to a DL reasoner
4. Return Xcerpt answers for which the DIG query is **"true"**

Eager interaction



Rule component Π:

r_1: price–in–usa(X,high) :–
made–by(X,Y),
American(Y),
monopoly–in–usa(Y,X).

r_2: price–in–usa(X,high) :–
made–by(X,Y),
European(Y).

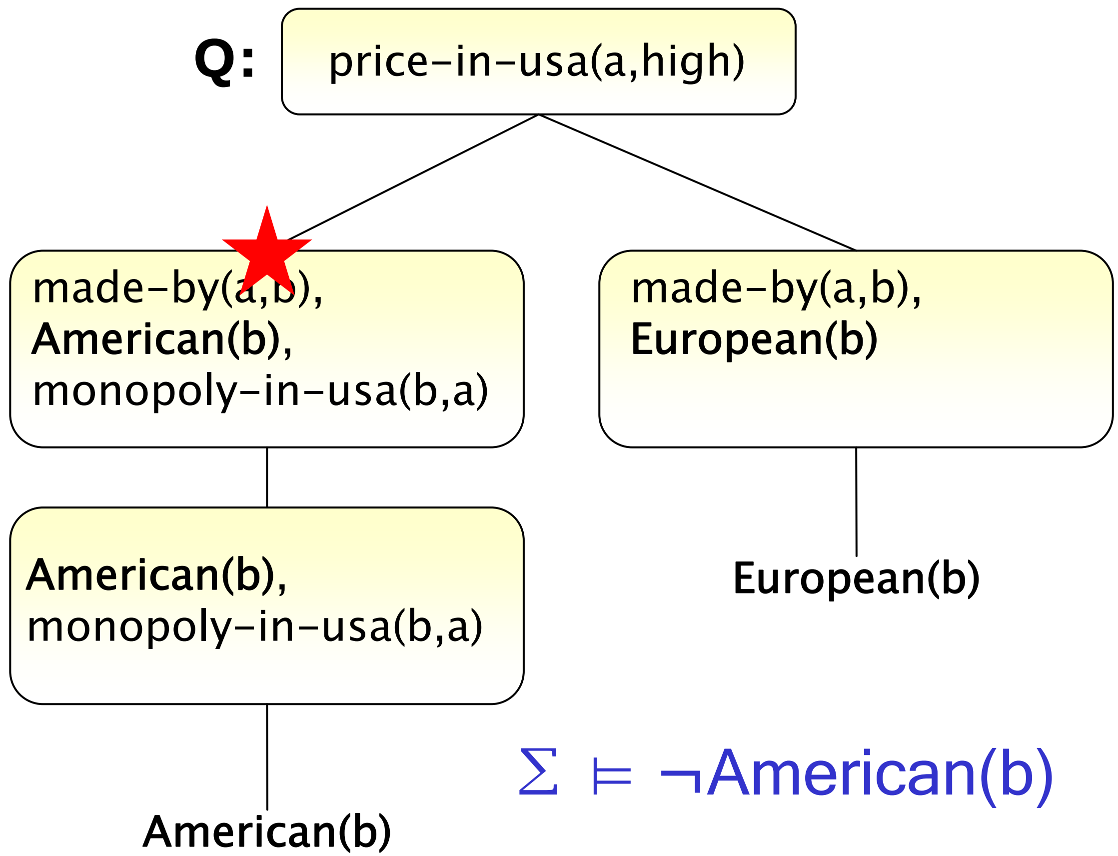
r_3: made–by(a,b).
r_4: monopoly–in–usa(b,a).

DL component Σ:

T–Box:
European ∩ American $\subseteq \bot$

A–Box: …
European(b)

Eager interaction



$\Sigma \models \neg$American(b)

➔ **Prune**

Rule component Π:

r_1: price-in-usa(X,high) :-
made-by(X,Y),
American(Y),
monopoly-in-usa(Y,X).

r_2: price-in-usa(X,high) :-
made-by(X,Y),
European(Y).

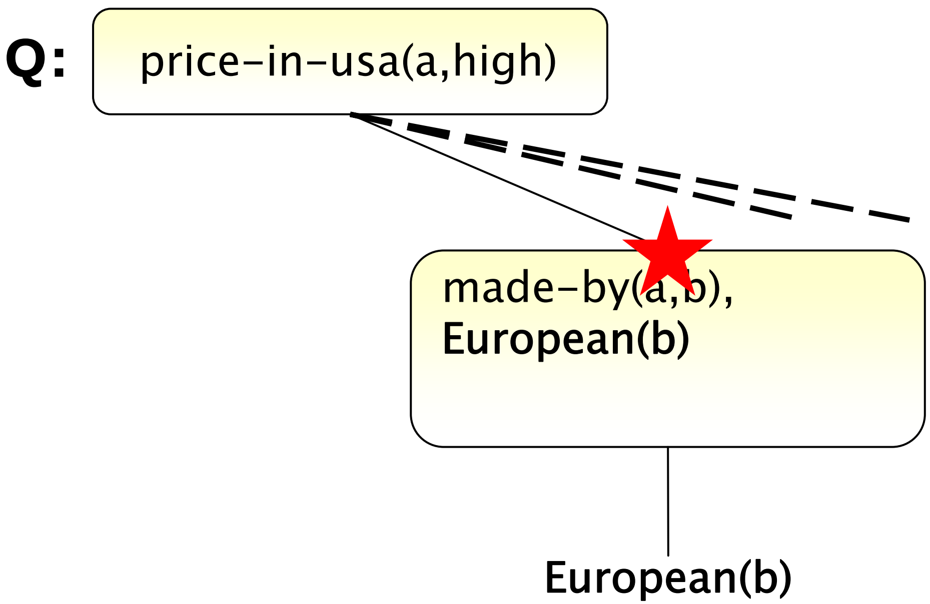
r_3: made-by(a,b).
r_4: monopoly-in-usa(b,a).

DL component Σ:

T-Box:
European $\cap$ American $\subseteq \bot$

A-Box: ...
European(b)

Eager interaction



$\Sigma \models$ European(b)

➔ **Prune**

Rule component Π:

r_1: price−in−usa(X,high) :−
made−by(X,Y),
American(Y),
monopoly−in−usa(Y,X).

r_2: price−in−usa(X,high) :−
made−by(X,Y),
European(Y).

r_3: made−by(a,b).
r_4: monopoly−in−usa(b,a).

DL component Σ:

T-Box:
European $\cap$ American $\subseteq \bot$

A-Box: …
European(b)

Prototype: Datalog + OWL DL

- Interfaces existsing reasoners
  - Rule reasoner: **XSB**
  - Ontology reasoner: **DIG compliant DL reasoner**
    - Only OWL concepts
    - Possible extension:
      - Allow roles in constraints through “rolling-up”
  - *Eager interaction*
    - No pruning yet…

  - Available at: **http://www.ida.liu.se/hswrl**

TECHNISCHE UNIVERSITÄT DRESDEN

Conclusions

- Combining general class of rules with constraints
  - Rules are negation-free, fixpoint semantics
- Non-logical rule-languages
  - E.g. Xcerpt
- Re-using existing reasoners
- Prototype integration
  - Datalog + OWL-DL
  - Using XSB + RacerPro

TECHNISCHE UNIVERSITÄT DRESDEN

Our work

- is motivated by and extends ***AL***-Log *[Donini et. al.]*
- aims at integration of existing reasoners
  - not restricted to Datalog
- supports reasoning by cases in DL, unlike:
  - **ASP+DL** *[Eiter et. al.]*
    - Handles negation, supports bi-directional flow of information between rules and DL KBs
- does not extend ontology languages like, e.g.:
  - **SWRL** *[Horrocks et. al.]*, **OWL-DL** *[Motik et. al.]*, **Safe Hybrid KBs** *[Rosati]*

TECHNISCHE UNIVERSITÄT DRESDEN

Future work

- How to re-use existing rule reasoners?
- Eager interaction
  - Practical use-cases and full implementation
- Other constraint languages
- Rules with negation
  - Well-founded semantics

The End

Thank you!